Introduction

“Pico, Fermi, Bagel” (also known as “Pico, Fermi, Zilch”) is a guessing game similar to Mastermind. The opponent (in this case, the computer) starts by picking an \( n \)-digit secret number, where \( n \) is an integer greater than zero. Then, we make guesses by typing in \( n \)-digit numbers until we’ve guessed the secret number correctly. The fewer guesses required, the better. The opponent provides some feedback after each move:

- “Fermi” is printed for each digit guessed correctly, in the correct place.
- “Pico” is printed for each digit guessed correctly, in the incorrect place.
- “Bagel” is printed if no digits are correct.

Game Details

To clear up some ambiguities about how the game works, read this section, but don’t worry: we’ve implemented the validation system for you. You’re welcome!

All “Pico” results are printed out before any “Fermi” results, and so the user does not know specifically which number is correct. For example, if the secret number is 250 and we guess 245, the feedback would be “Pico Fermi.”

The winning guess should receive “Fermi” \( n \) times. For example, if the secret number is 921 and we guess 921, the feedback would be “Fermi Fermi Fermi.”

Note that “Fermi” and “Pico” may show up multiple times (e.g. a secret of 123 and a guess of 231 produce “Pico Pico Pico”) but “Bagel” is only printed once if necessary. For example, a secret of 789 and a guess of 123 produce “Bagel.”

Also, because some may not consider leading zeros to be digits (e.g. would you consider 000–099 to be three-digit numbers?), we will make the opponent only choose secret numbers between \( 10^{n-1} \) and \( 10^n-1 \). This will be your responsibility to implement, but it will likely make other parts of your code easier.

Implementation Details

The design of this project has already been done for you, so all you need to do is implement it. You will implement three classes and make use of a fourth, which is provided, to solve this problem in a modular fashion. Use the UML diagram and documentation provided in the following sections to understand how the project is designed and how you should implement each method. You must adhere to this design exactly! This includes variable names, method names, argument types, return types, etc. Also, DO NOT program any constructors for the Player and Engine classes! Instead, we will use Java’s default constructor to create new Player objects and new Engine objects.

Hopefully, you will find this way of programming easier to work with, since you can write and test your code in smaller, organized blocks, and better reuse code. You may even think of each method as a smaller programming project!

You will need to use everything you’ve learned so far to complete this project, including loops, arrays, classes, objects, and methods. You may also use the following classes provided by Java at some point (though, not required):

- \( \text{int} \ \text{nextInt}(\text{int} \ n) \), an instance method in the Random class
  - Returns a number between 0 (inclusive) and \( n \) (exclusive). Hint: you can use simple mathematical operations to specify a different lower bound than 0.
- \( \text{double} \ \text{pow}(\text{double} \ a, \ \text{double} \ b) \), a static method in the Math class
  - Returns the value of \( a^b \).
- \( \text{int} \ \text{MAX VALUE} \), a static field in the Integer class
  - The largest value an integer can have (\( 2^{31}-1 \)).
- \( \text{int} \ \text{parseInt} \), a static method in the Integer class that converts a String input to an integer.

Though you are welcome to do so, you are not required to account for invalid user input, including digit values that are too big and guesses that do not have the correct number of digits.
Documentation

Class: Bagels – The “driver” class that users will execute to play the game.

Methods

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void</td>
<td>main(String[])</td>
<td>Called automatically when the program is run. It should run according to the flowchart below and the examples at the end of this project. This method may contain a Scanner object to retrieve the number of digits and the name from a user.</td>
</tr>
</tbody>
</table>

Class: Player – Models the data and actions associated with the person playing the game. **DO NOT** program any constructors for this class.

Fields

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Name</td>
<td>The first name of the player.</td>
</tr>
<tr>
<td>int</td>
<td>fastestWin</td>
<td>Like a high score. How quickly was the player able to guess the number? Best score possible is 1.</td>
</tr>
<tr>
<td>int</td>
<td>gamesCompleted</td>
<td>The number of games that have been completed (and thus won).</td>
</tr>
<tr>
<td>Scanner</td>
<td>keyboard</td>
<td>Used for user input in askForGuess(). This should be initialized once and reused for each call to askForGuess().</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>askForGuess()</td>
<td>Using the keyboard field, the guess is read in from the keyboard as a String.</td>
</tr>
<tr>
<td>String</td>
<td>getName()</td>
<td>Returns the player’s name.</td>
</tr>
<tr>
<td>int</td>
<td>getFastestWin()</td>
<td>Returns the fastest win.</td>
</tr>
<tr>
<td>int</td>
<td>getGamesCompleted()</td>
<td>Returns the number of games completed.</td>
</tr>
<tr>
<td>void</td>
<td>setName(String)</td>
<td>Sets the player’s name.</td>
</tr>
<tr>
<td>void</td>
<td>updateStats(int)</td>
<td>Should be called once after finishing each game. It updates the gamesCompleted field and possibly the fastestWin field.</td>
</tr>
</tbody>
</table>
Class: Engine – Encapsulates data and actions related to the actual game in progress. **DO NOT** program any constructors for this class.

**Fields**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>numDigits</td>
<td>The number of digits the user wishes to play with.</td>
</tr>
<tr>
<td>int[]</td>
<td>secretNumber</td>
<td>The computer’s secret number.</td>
</tr>
<tr>
<td>Random</td>
<td>randomNumberGenerator</td>
<td>An instance of Java’s Random class, used to generate pseudorandom secret numbers.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int[]</td>
<td>convertNumToDigitArray(String)</td>
<td>Converts a number (as a String) to an array of ints. The ordering <strong>must</strong> go from most to least significant digits. For example, if the number String is “732” then index 0 of the array should contain 7.</td>
</tr>
<tr>
<td>int</td>
<td>getNumDigits()</td>
<td>Returns the number of digits.</td>
</tr>
<tr>
<td>int[]</td>
<td>getSecretNumber()</td>
<td>Returns the secret number.</td>
</tr>
<tr>
<td>void</td>
<td>generateNewSecret()</td>
<td>Changes the secretNumber field to a random number in the range 10^n-1 and 10^n-1, and this method will need to be invoked each time a game is started.</td>
</tr>
<tr>
<td>void</td>
<td>setNumDigits(int)</td>
<td>Sets the number of digits to the input parameter.</td>
</tr>
<tr>
<td>void</td>
<td>setSecretNumber(int[])</td>
<td>Sets the values in this object’s secretNumber to a copy of the values in the secretNumber input parameter, and the order of values must be the same in this object’s secretNumber and the secretNumber input parameter. This method should call the setNumDigits method to update the number of digits based on the input parameter. <strong>Note: This method may or may not be used in your final program. Please implement it anyway. We need it when grading.</strong></td>
</tr>
</tbody>
</table>

Class: Validator – Contains a single static method to validate guesses. This class is provided and should not be modified.

**Methods**

<table>
<thead>
<tr>
<th>Type</th>
<th>Method Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static boolean</td>
<td>validateGuess(int[], int[], int)</td>
<td>Prints out the feedback from the opponent (e.g. “Fermi Pico”) and returns <strong>true</strong> if the guess was correct or <strong>false</strong> if the guess was incorrect. The last argument (numDigits) should match the length of both arrays.</td>
</tr>
</tbody>
</table>
Flowchart

Study the flowchart (in addition to the sample runs) below when writing your main method.

```
Start

Welcome

Enter number of digits

Enter player name

Starting game

Enter guess

Print feedback

Wrong

Menu

Quit

Right

Congratulations

p r q
```
Additional Instructions

1. Your program must include a comment between the import statements and the class declaration for each class file you are submitting. Copy and agree to the comments below, and fill in the file name, your name, the submission date, and the program’s purpose. Five points will be deducted if this step is not completed for every class file you submit.

    /*
     * [Class name here].java
     * Author:  [Your name here]
     * Submission Date:  [Submission date here]
     * Purpose: A brief paragraph description of the program. What does it do?  How does it do it?
     * Statement of Academic Honesty:
     * The following code represents my own work. I have neither received nor given inappropriate assistance. I have not copied or modified code from any source other than the course webpage or the course textbook. I recognize that any unauthorized assistance or plagiarism will be handled in accordance with the University of Georgia’s Academic Honesty Policy and the policies of this course. I recognize that my work is based on a programming project created by the Department of Computer Science at the University of Georgia. Any publishing of source code for this project is strictly prohibited without written consent from the Department of Computer Science.
     */

2. All methods must be commented, and comments for each method must include the purpose of the method, the input parameters, and what the method returns. Any precondition or postcondition assumptions associated with a method must be mentioned in its comments. Five points will be deducted if this step is not completed for every method in every file you submit for this project.

3. All variables must be named well and conform to Java standards as discussed during lecture class and shown in the course textbook. Instance variables for this project must be named exactly as stated, and they are already named well and conform to Java standards. Five points will be deducted if your source code contains poorly named variables.

4. All source code must be indented properly and consistently. Check your course textbook or ask your lab instructor how to indent your source code properly. Five points will be deducted if your source code is not indented properly or its indentation is inconsistent.

5. After implementing all classes and methods aforementioned, you should place the file BagelsMethodCheck.java in the same source directory (folder) as the rest of your class files for this project. You should run the main method in BagelsMethodCheck.java to make sure your program follows the design specifications exactly as indicated. This class checks that your methods are declared properly. It does NOT check that your methods are correct (it is your job to test that each method works correctly).

All instructions must be followed for full credit to be awarded. If you have any questions about the requirements, then ask your lab instructor days before the assignment is due.

Project Submission

After you have thoroughly tested your program with all of the examples provided plus your own examples, submit Bagels.java, Player.java, and Engine.java to eLC for grading. You should not include Validator.java or any .class files.
Project Grading

All projects are graded out of a possible 100 points. Programs can be submitted up to 48 hours late, but late programs will lose points per the syllabus. Programs not submitted within 48 hours after the deadline will receive a grade of zero. Programs that do not compile will also receive a grade of zero. You must make absolutely certain your program compiles before submitting, and you must thoroughly test your program with different inputs to verify that it is working correctly. You must make absolutely certain your project conforms to all UML specifications; otherwise, it may not compile or run correctly with our testing program(s), which may result in a failing grade on this project.

This project will be graded for both correctness and adherence to stated instructions:

Correctness [100pts]

- 100 points for correct output on various test cases.

Not following instructions (poor programming style, etc.)

- Points will be deducted for not following instructions. If your program does not compile because you failed to follow instructions, then you may receive a failing grade on this assignment. In computer science, it is extremely important to follow technical specifications.

Example Runs

Your program should be consistent with the examples below and all specifications in this project. All input/output should be formatted as shown.

Welcome!
Enter the number of digits to use: 2
Enter the player's name: Kyle

Starting game #1.
Enter your guess: 12
Bagel
Enter your guess: 34
Bagel
Enter your guess: 56
Fermi
Enter your guess: 57
Fermi
Enter your guess: 58
Fermi
Enter your guess: 59
Fermi Fermi
Congratulations! You won in 6 moves.

Statistics for Kyle:
Games completed: 1
Number of digits: 2
Fastest win: 6 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? p
Starting game #2.
Enter your guess: 83
Bagel
Enter your guess: 72
Fermi
Enter your guess: 92
Fermi Fermi
Congratulations! You won in 3 moves.

Statistics for Kyle:
Games completed: 2
Number of digits: 2
Fastest win: 3 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? r

Enter the number of digits to use: 3
Enter the player's name: Gertrude

Starting game #1.
Enter your guess: 123
Pico
Enter your guess: 456
Pico
Enter your guess: 789
Pico
Enter your guess: 147
Pico
Enter your guess: 258
Pico Fermi
Enter your guess: 618
Fermi Fermi
Enter your guess: 628
Fermi
Enter your guess: 418
Fermi Fermi
Enter your guess: 218
Fermi Fermi
Enter your guess: 118
Fermi Fermi
Enter your guess: 418
Fermi Fermi
Enter your guess: 318
Fermi Fermi
Enter your guess: 518
Fermi Fermi Fermi
Congratulations! You won in 13 moves.

Statistics for Gertrude:
Games completed: 1
Number of digits: 3
Fastest win: 13 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? q

Goodbye!

Welcome!
Enter the number of digits to use: 3
Enter the player's name: Fanny

Starting game #1.
Enter your guess: 182
Pico
Enter your guess: 314
Fermi
Enter your guess: 516
Fermi Fermi
Enter your guess: 517
Pico Fermi
Enter your guess: 716
Fermi Fermi Fermi
Congratulations! You won in 5 moves.

Statistics for Fanny:
Games completed: 1
Number of digits: 3
Fastest win: 5 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? p

Starting game #2.
Enter your guess: 258
Bagel
Enter your guess: 149
Bagel
Enter your guess: 367
Pico
Enter your guess: 700
Fermi
Enter your guess: 607
Fermi Fermi
Enter your guess: 606
Fermi Fermi Fermi
Congratulations! You won in 6 moves.

Statistics for Fanny:
Games completed: 2
Number of digits: 3
Fastest win: 5 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? q

Goodbye!

--------------------------------------------------------------------
Welcome!
Enter the number of digits to use: 1
Enter the player's name: Uma

Starting game #1.
Enter your guess: 5
Bagel
Enter your guess: 1
Bagel
Enter your guess: 9
Bagel
Enter your guess: 3
Bagel
Enter your guess: 8
Bagel
Enter your guess: 8
Bagel
Enter your guess: 2
Bagel
Enter your guess: 4
Bagel
Enter your guess: 7
Bagel
Enter your guess: 6
Fermi
Congratulations! You won in 10 moves.

Statistics for Uma:
Games completed: 1
Number of digits: 1
Fastest win: 10 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? q

Goodbye!